Aplicación basada en Flutter para el control de dispositivos BLE

Victor Merino Rueda

Resum— En los últimos años, el uso de herramientas multiplataforma ha ido en aumento. De entre ellas, la demanda de aplicaciones desarrolladas en Flutter ha crecido. Idneo, empresa donde realizo las prácticas, ha querido ganarse un sitio en este mercado buscando con la realización de este proyecto establecer una base de conocimientos sobre esta tecnología, centrándose en funcionalidades de Bluetooth Low Energy (BLE), considerando el gran uso futuro de estas. Para ello, se ha planteado la realización de una prueba de concepto que permita mostrar diversas funcionalidades como la gestión de dispositivos, navegación entre pantallas, aplicación multilenguaje, entre otras. Dotándola a su vez, de funcionalidades BLE como la búsqueda de dispositivos cercanos, el descubrimiento de servicios, características y descriptores; así como la lectura y escritura de estos. El resultado final es una aplicación que permite mostrar los conocimientos obtenidos, pudiendo entrar en un nuevo sector de negocio como son las aplicaciones móviles.

Paraules clau— Desarrollo de aplicaciones multiplataforma, Flutter, Dart, Android, IOS, Desktop, Web, Conectividad Bluetooth, BLE, PoC

Abstract— In the last few years, the use of cross-platform tools has been rapidly increasing. Among all of them, the demand for applications developed in Flutter has particularly grown. Idneo, the company where I am currently doing an internship in, has dedicated efforts and resources at earning a stake in this market in recent years. The execution of this project has aimed at establishing a solid knowledge base on this technology, focusing on Bluetooth Low Energy (BLE) functionalities, and considering the enourmous future potential of these. In order to carry out this project, a proof of concept has been proposed so as to show the different actual applications that cross-platform tools developed in Flutter may have (such as device management, navigation between screens, multilanguage application, among others), while at the same time providing it with BLE functionalities (for instance, the search for nearby devices, the finding of new services, characteristics and descriptors, as well as the reading and writing of these). The final result is an application that allows showing the knowledge acquired during the development of the project, which at the same time can be effectively used to enter a new business sector (such as mobile applications).

Index Terms— Cross platform software development, Flutter, Dart, Android, IOS, Desktop, Web, Bluetooth Connectivity, BLE, PoC

1 Introducción

Siendo Flutter una tecnología relativamente nueva, muchas son las aplicaciones que han optado por ésta desde su lanzamiento en mayo de 2017. El framework desarrollado y mantenido por Google está ganando cada vez más adeptos [1]. Idneo [2], empresa interesada en el aprendizaje e inclusión de esta tecnología entre sus servicios, es la empresa en la cual estoy realizando prácticas extracurriculares. Tal y como se autodefina la misma: "Es una empresa global de diseño estratégico e ingeniería, que apoya a los clientes a transformar su negocio a través de la tecnología".

Debido a ello, es normal ver que no quiera quedarse atrás en el uso de este framework, adaptándose así a la gran cantidad de empresas que están ofreciendo el desarrollo de aplicaciones multiplataforma (iOS/Android principalmente) entre sus sectores de negocio.

Haciendo referencia a lo comentado en el párrafo anterior, uno de los principales motivos por los que Idneo quiere incorporar esta tecnología es la creciente demanda de proyectos que la utilizan por parte de clientes actuales de la empresa. Es por eso por lo que se pretende apostar por ésta, satisfaciendo las actuales solicitudes, y expandiendo dicho servicio entre otros posibles clientes.

Creo necesario contextualizar un poco mi perfil académico para poder entender mejor la motivación personal que me empuja a realizar dicho proyecto. Previamente a la realización de la mención de Ingeniería del Software dentro del Grado en Ingeniería Informática de la UAB dentro de la cual se está llevando a cabo este TFG, tuve el placer de cursar un Grado Superior de Desarrollo de Aplicaciones Multiplataforma (DAM).

Hace cuatro años que finalicé el Grado Superior, pero claramente marcó el rumbo que posteriormente seguiría dentro del gran número de ramas de las que dispone el sector de la informática. Es por ello por lo que no debe sorprender a nadie el gran interés que presento a la hora de conocer una nueva tecnología con la que desarrollar software, y más aún, si hablamos de una tecnología puntera y respaldada por, nada más y nada menos, que Google.

El hecho de no disponer de ningún perfil técnico en este sector en la empresa en la cual estoy realizando prácticas, me empuja aún más a querer sumergirme en el mundo de Flutter y poder perfilarme como un trabajador de referencia en la empresa en dicho sector.

Como se ha comentado anteriormente, el uso de este framework es completamente nuevo en Idneo por lo que uno de los principales objetivos que se quieren cubrir con la realización de este proyecto es la inclusión de esta nueva tecnología entra las que ya se ofrecen por parte de la empresa. Para llevar a cabo dicha inclusión, se realizará la formación necesaria sobre el lenguaje Dart y Flutter para poder empezar a desarrollar una pequeña prueba de concepto (PoC) [3], la cual permitirá mostrar a los clientes qué tipo de producto se les puede ofrecer. Aprovechando dicho período de formación, se buscarán y adaptarán como metodología de trabajo para futuros proyectos las mejores prácticas de programación y de arquitectura sobre estas tecnologías.

Otro de los objetivos principales es la investigación e implementación de la conectividad Bluetooth Low Energy (BLE) [REF] de forma nativa para cada tipo de dispositivo. Esta funcionalidad ha sido requerida para posibles futuros proyectos, por lo que se quiere dotar a la aplicación que se desarrollará de ésta. Se ha considerado desde la empresa que se deben analizar dos de los plugins existentes que dotan con un módulo a Flutter para realizar dicha conectividad BLE. Es por eso por lo que se ha decidido crear un subproyecto en Flutter que permita integrar cada puglin disponible para disponer de un proyecto plantilla (template) que sirva como ejemplo de uso para futuros proyectos que utilicen conectividad BLE.

Finalmente hay que aclarar que, con todo lo descrito anteriormente, se ha procedido a desarrollar una PoC de un posible proyecto entrante, consistente en la migración de una aplicación ya existente a Flutter. Dicha aplicación permite el control de robots de limpieza de piscinas. Como parte final de esta PoC se integrará el subproyecto mencionado anteriormente para dotar a esta prueba de concepto con conectividad BLE.

2. Estado del arte

Después de poco más de tres años desde su lanzamiento, la comunidad de Flutter ha crecido en tamaño y popularida. Es por ello por lo que, junto a la gran documentación aportada por el equipo de desarrollo de Google que se encarga del mantenimiento de este framework, es mucho el soporte que se puede encontrar por parte de los desarrolladores que utilizan esta tecnología. Dicho soporte puede verse reflejado en forma de ayudas en foros, tutoriales, entre otros. Pese a ello, no existe documentación oficial de buenas prácticas que seguir sobre la arquitectura a la hora de desarrollar una aplicación en el momento en que se realiza este documento. Si podemos encontrar, sin embargo, dicha documentación sobre el lenguaje Dart [4].

En la actualidad pueden encontrarse grandes empresas que poseen aplicaciones móviles de renombre desarrolladas con esta tecnología como puede ser el caso de BMW, Alibaba, Tencent, o Google [5]. Otra gran compañía que cuenta con una aplicación desarrollada en Flutter es eBay. Hablamos de su aplicación *eBay Motors app* [6], la cual se utilitza para la compra y venta de vehículos y de la cual podemos ver una pequeña muestra en la figura 1.

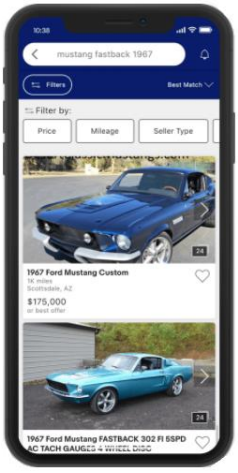

Figura 1: *Captura de pantalla de eBay Motors app.*

Como se ha mencionado con anterioridad, actualmente existe una aplicación que controla el uso de dichos robots de limpieza de piscinas. Esta aplicación permite conectarse al módulo de carga de los robots mediante BLE y transferir datos hacia dicha aplicación. Esta muestra información adicional al usuario como el tiempo actual, el PH de la piscina, entre otros. Con la migración de ésta se pretende simplificar su mantenimiento, ya que pasaría a utilizar un único código fuente y se actualizaría la tecnología empleada, con las ventajas que ello conlleva. Entre estas ventajas encontraríamos la mejor adaptación a nuevas versiones de los SO, posibilidad de utilizar nuevas funcionalidades, mejora de la parte visual, entre otras.

3. Requisitos

3.1.- Requisitos Hardware
Especificaciones del ordenador empleado:
Intel Core i5-5200U, 8GB RAM, 240GB SSD, Windows 10.
Otros dispositivos:
Smartphone Android con SO Android 10 (se utilizarán

emuladores para testear la aplicación en diferentes dispositivos).

3.2.- Requisitos Software

A continuación, se listan las diferentes herramientas software que se usarán para desarrollar el proyecto:

- **Dart:** Dicho lenguaje de programación es la base sobre la que trabaja el framework Flutter y su conocimiento es esencial para desarrollar aplicaciones multiplataforma con dicho framework.
- **Flutter:** Flutter es un framework que proporciona las herramientas necesarias para desarrollar aplicaciones nativas en IOS, Android, Web y, recientemente, también para Desktop. A partir del uso de widgets, permite la creación de screens completamente personalizables dando la capacidad de modelar cada píxel de la pantalla.
- **Visual Studio Code:** Es el IDE (Integrated Development Environment) seleccionado para desarrollar el código de la aplicación Flutter. Ha sido seleccionado entre los posibles candidatos, como podría ser Android Studio, por su ligereza y su previo conocimiento sobre el uso de este [7].
- **Git:** Empleado para el control de versiones, este software nos permitirá llevar un control del código desarrollado [8].
- **BitBucket:** Este servicio de alojamiento web proporciona acceso al control de versiones del proyecto, manteniendo el código en la nube, facilitando el acceso a esto [9].
- **Android Studio:** Pese a no haber sido seleccionado como IDE de desarrollo, Android Studio brinda la posibilidad de emular una gran variedad de dispositivos Android, permitiendo correr la aplicación a desarrollar en estos [10].
- **Firebase:** Dentro de la plataforma desarrollada por Google para el desarrollo de aplicaciones móviles, ésta permite la creación de un back-end sencillo de utilizar permitiendo el uso, por ejemplo, de una base de datos en tiempo real. Dicha plataforma permite la focalización sobre la parte front-end del proyecto, objetivo principal de este [11].

3.3.- Requisitos funcionales

- [RF.1]: El sistema tiene que permitir al usuario conocer el estado del PH de la piscina.
- [RF.2]: El sistema tiene que permitir al usuario conocer el estado de la temperatura de la piscina.
- [RF.3]: La aplicación tiene que ser capaz de mostrar el estado actual de los diferentes robots de piscina asociados.
- [RF.4]: La aplicación tiene que permitir añadir los dispositivos (robots) vinculados a ésta.
- [RF.5]: La aplicación tiene que permitir borrar los dispositivos (robots) vinculados a ésta.
- [RF.6]: La aplicación tiene que permitir editar los dispositivos (robots) vinculados a ésta.
- [RF.7]: El sistema tiene que permitir al usuario conectarse mediante Bluetooth con el módulo de carga de los robots en dispositivos Android.
- [RF.8]: La aplicación permitirá al usuario modificar la hora del día en que el robot de limpieza se activará.
- [RF.9]: La aplicación permitirá al usuario modificar el día de la semana en que el robot de limpieza se activará.
- [RF.10]: El sistema ha de permitir la edición de las preferencias de limpieza que el robot emplea.
- [RF.11]: El sistema permitirá al usuario conectarse mediante Bluetooth con el módulo de carga de los robots en dispositivos IOS.
- [RF.12]: El sistema ha de dejar modificar el idioma entre español, inglés y catalán.

El diagrama de casos de uso correspondiente a estos requisitos funcionales se muestra en la Figura 2.

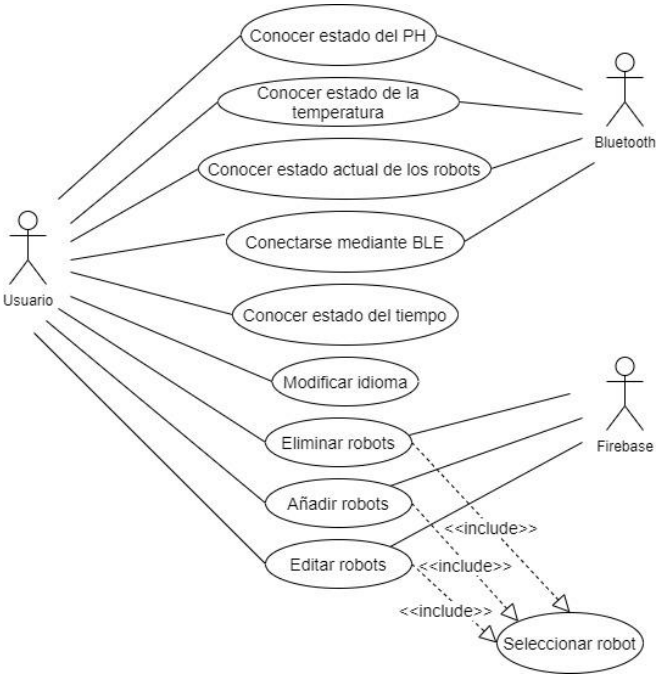

Figura 2: *Diagrama de casos de uso SPOCA.*

3.4.- Requisitos no funcionales

- [RNF.1]: El sistema debe desarrollarse aplicando buenas prácticas de programación recomendadas por Dart.
- [RNF.2]: El sistema debe desarrollarse aplicando patrones de arquitectura recomendados por Flutter.
- [RNF.3]: La aplicación móvil debe conectarse al módulo de carga a una distancia mayor a 5 metros mediante el uso de Bluetooth.
- [RNF.4]: La aplicación deberá tener conexión a internet en todo momento.
- [RNF.5]: La aplicación debe utilizar elementos visuales estandarizados para la UI según su sistema operativo (SO).
- [RNF.6]: El sistema tiene que estar desarrollado para plataformas Android.

- [RNF.7]: El sistema deberá estar desarrollado para plataformas IOS.
- [RNF.8]: El sistema tiene que contar con pruebas funcionales
- [RNF.9]: El sistema tiene que poder transmitir datos bidireccionalmente con el módulo de carga de los robots en dispositivos Android.
- [RNF.10]: El sistema podrá transmitir datos bidireccionalmente con el módulo de carga de los robots en dispositivos IOS.

3.5.- Restricciones

- El proyecto se debe realizar con el siguiente equipo: Intel Core i5-5200U, 8GB RAM, 240GB SSD, Windows10.
- La realización del proyecto no debe suponer coste alguno aparte del salario del trabajador.
- El proyecto tiene que realizarse con software libre.
- Se tiene que utilizar la metodología ágil Kanban para el desarrollo del proyecto [12].

4. RIESGOS DEL PROYECTO

- **[RP.1] Falta de planificación y de seguimiento:**
- Imposibilidad de cumplir con el plan establecido debido a la mala la gestión del proyecto.
- **[RP.2]Falta de compromiso y/o de apoyo:**
- Ausencia de interés por alguna de las principales partes interesadas provocando una mala comunicación sobre el proyecto a desarrollar y lo que esto conlleva.
- **[RP.3] Mayor esfuerzo que el estimado:**
- Necesidad de un esfuerzo superior al esperado provocando retrasos o imposibilitando el desarrollo de todas las funcionalidades.
- **[RP.4] El producto final no es el esperado:**
- Malestar en el cliente (Idneo) provocado por la falta de aproximación entre el producto esperado y el recibido.
- **[RP.5] Falta de conocimientos:**
- Carencia de conocimientos sobre las teconologías utilizadas ocasinando retrasos o partes incompletas en la aplicación.
- **[RP.6] Ruptura del equipo de desarrollo:**
- Incapacidad de seguir desarrollando el proyecto debido al deterioro del equipo utilizado.
- **[RP.7] Pérdida de código:**
- Pérdida parcial o total de partes del SW generado.

En la Tabla1 se muestra la probabilidad de ocurrencia e impacto asociado de cada uno de los riesgos. En la Tabla 2 se describen los planes de contingencia diseñados para combatir dichos riesgos.

Riesgo	Probabilidad	Impacto
[RP.1]	Media	Medio
[RP.2]	Nula	Medio
[RP.3]	Media	Medio
[RP.4]	Nula	Crítico
[RP.5]	Media	Medio
[RP.6]	Nula	Crítico
[RP.7]	Nula	Crítico

Tabla1: *Probabilidad/Impacto de los riesgos*

Riesgo	Contingencia
[RP.1]	Gestionar de forma correcta las diferentes fases y tareas del proyecto empleando herramientas pensadas para ello, así como la organización de reuniones semanales para comprobar el cumplimiento del schedule marcado.
[RP.2]	Seleccionar cuidadosamente los diferentes stakeholders conociendo de antemano el nivel de compromiso que estos pueden llegar a aportar al proyecto.
[RP.3]	Contactar con personas expertas en la gestión del desarrollo de SW, así como con especialistas en tecnologías iguales y/o semejantes, para realizar una buena estimación del esfuerzo necesario a emplear.
[RP.4]	Recopilar exhaustivamente los diferentes requisitos. Prestar mucha atención a las solicitudes del cliente al inicio y durante el proyecto, comprendiendo las necesidades que este tiene y lo que se desea recibir.
[RP.5]	Implicarse de forma directa en el aprendizaje de las diferentes tecnologías utilizadas, a partir de la documentación existente.
[RP.6]	Adquirir una nueva unidad que pueda utilizarse en caso de necesidad.
[RP.7]	Utilizar heramientas de control de versión y software en la nube que permita el uso y restauración de este en caso de necesidad en diferentes dispositivos.

Tabla2: *Contingencia ante cada riesgo*

5. PLANIFICACIÓN

- **Fase 1 – Formación (50 horas):** Período de tiempo establecido para el aprendizaje básico de la tecnologia Flutter y el lenguaje de programación Dart.
- **Fase 2 – Estrategia (60 horas):** Investigación de las majores prácticas para aquitecturas basadas en Flutter para apps moviles multiplataforma. Análisis de estrategías de control de hardware nativo con Flutter mostrando especial interés en la conectividad Bluetooth.
- **Fase 3 – Diseño (20 horas):** Propuesta de UI/UX para la PoC a desarrollar.

- **Fase 4 – Desarrollo (145 horas):** Implementación de las diferentes pantallas de la app móvil siguiendo los conocimientos previamente adquiridos de buenas prácticas y arquitectura. Conjuntamente, dotar de las las diferentes funcionalidades básicas a dicha app.
- **Fase 5 – Entrega (15 horas):** Fase final de proyecto donde procederá a desplegarse la aplicación realizada para los diferentes sistemas operativos, así como la finalización de los diferentes documentos.

En el apéndice 1 se muestra el diagrama de Gantt que representa la distribución temporal de las tareas del proyecto.

6. METODOLOGÍA

Respecto a la metología implementada en la realización de este proyecto, como bien se especifica en los requisitos no funcionales, se ha decidido utilizar Kanban. Se ha apostado por la realización semanal de dos reuniones de retroalimentación, una con el tutor responsable de Idneo y otra con el tutor responsable de la universidad. Dicha metodología tiene como base el desarrollo incremental e iterativo, tiniendo en cuenta que los requisitos y las soluciones evolucionan en el tiempo según las necesidades específicas del proyecto.

7. DESARROLLO

7.1.- Aprendizaje de Flutter

El material empleado a la hora de realizar este aprendizaje ha sido variado, pero hay que destacar que principalmente se ha utilizado la documentación oficial de Flutter que versa sobre los widgets y otros aspectos importantes. Para canalizar y asumir estos conceptos, y a su vez ampliar los conocimientos adquiridos, se ha seguido el curso de Udemy *Flutter & Dart - The Complete Guide [2020 Edition]* [13].

Otro aspecto importante que comentar en este apartado es la búsqueda y adaptación de las mejores prácticas de programación y de arquitectura sobre las tecnologías Dart y Flutter.

A continuación, se hace una pequeña explicación de los dos documentos que han permitido realizar lo descrito anteriormente:

- **Best Practices**

Este documento pretende recoger las mejores prácticas a la hora de desarrollar proyectos en Flutter y Dart.

- **Guía de programación**

Este documento tiene como objetivo recoger algunos de los aspectos más importantes que se han adquirido durante el periodo de formación, aun que hay que tener en cuenta que intenta evitar redundancias sobre un mismo tema con el anterior documento (Best Practices).

7.2.- Conectividad BLE

Uno de los objetivos principales de este proyecto es el aprendizaje sobre la conexión bluetooth en Flutter. Finalmente se ha decidido optar por el uso en este proyecto de la conectividad Bluetooth Low Energy (BLE). Se ha considerado la mejor opción entre esta y la conectividad standard de Bluetooth, debido principalmente al hecho de que la mayoría de los dispositivos de Internet of Things (IoT) la emplean.

Se prevé que este tipo de dispositivos sean los que acaparen la mayoría de los proyectos venideros que necesiten una aplicación en Flutter dentro de la empresa. Otro punto a favor para su uso ha estado que el robot que se prevé utilizar en uno de los proyectos que se realicen a corto plazo utiliza dicha tecnología.

De esta forma, desde Idneo se pretende disponer de unas bases sobre el tema para poder ofrecerlo como aliciente a los clientes que están interesados en el desarrollo de una aplicación móvil con dicha característica.

7.2.1.- Dispositivo BLE

Se pretende desarrollar un subproyecto que sirva como base para futuros proyectos venideros que necesiten de conectividad BLE. Para ello, de ahora en adelante llamaremos a este subproyecto como *BLE_Template*. Para realizar las diferentes pruebas de conexión que se van a realizar en este proyecto, se ha decidido utilizar el dispositivo PAN1780 de Panasonic del cual se puede ver una foto en la figura 3 [14].

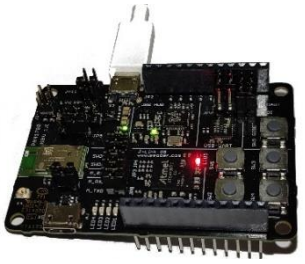

Figura 3: *Foto del dispositivo PAN1780 de Panasonic.*

7.2.2.- BLE_Template

En esta sección se da una visión general del mencionado subproyecto BLE_Template. En esta parte del proyecto, se ha procedido a realizar otro proyecto en Flutter que contiene los componentes básicos que permiten establecer una conexión BLE con otro dispositivo.

Desde Idneo se ha creído conveniente crear un proyecto *template* que se adapte al uso de diferentes plugins, para determinar así cuál es más conviente utilizar por la empresa. Para ello se ha procedido a analizar dos plugins existentes en el repositorio oficial que Dart y Flutter ofrecen. Pese a estar en dicho repositorio, estos plugins han sido realizados por personas externas a estas organizaciones, por lo que se requiere estudiar cuál de ellos utilizar en los proyectos: **flutter_reactive_ble** [15] y **flutter_blue** [16]. Los requisitos de dicha aplicación base se detallan en el apéndice 2.

La estructura del software que se utiliza es la misma que se emplea para la aplicación de SPOCA, la cual se puede encontrar en el apartado 7.3.2.

7.2.3.- Análisis de plugins

Para llevar a cabo dicho análisis se ha empezado por desarrollar las diferentes vistas y widgets, así como la estructura interna del código que permite el uso de cada plugin teniendo que hacer el mínimo de cambios posibles para adaptar cada uno de estos. Para ello, se ha creado un repositorio en Bitbucket que contiene dicho proyecto, y se han creado dos forks de este proyecto principal donde se ha adaptado cada puglin en uno de estos.

La creación de este proyecto base ha presentado dificultades debido a las grandes diferencias que existen a la hora de adaptar las funcionalidades de BLE en cada plugin.

La adaptación de los dos plugins se ha llevado a cabo con éxito. Pese a ello, antes de poder completar todas las funcionalidades que el plugin **flutter_reactive_ble** permite, se ha decidido por parte de todos los integrantes que llevan a cabo este proyecto (tutor de empresa, tutor de la universidad y estudiante), no continuar con la adaptación de este. Esto es debido a los grandes esfuerzos que se han estimado que llevaría hacerlo.

En lugar de analizar y comparar finalmente los dos plugins en base a unas KPI's, se ha decidido adaptar el plugin **flutter_blue** y ver si se adapta a las necesidades de la empresa.

7.3.- Swimming Pool Cleaning Assistant

El siguiente apartado contiene las diferentes partes que constituyen el proyecto general denominado Swimming Pool Cleaning Assistant (SPOCA). Como se ha comentado anteriormente, se pretende realizar una prueba de concepto (PoC) de una aplicación que controla y gestiona unos robots de limpieza que se conectan a la base de carga la cual se puede comunicar con nuestra aplicación mediante BLE.

En los siguientes apartados se pueden encontrar el mapa de pantallas planteado, la estructura de proyecto, entre otros.

7.3.1.- Mapa de pantallas

Las diferentes pantallas de la aplicación final utilizan la siguiente estrcutura como guía:

- Menú inferior:
  - Tab1: Inicio
  - Tab2: Limpieza
  - Tab3: Dispositivos
    - Detalles de dispositivo
    - Agregar dispositivo
- Menú lateral:
  - Idioma
  - Configuración de piscina

7.3.2.- Arquitectura PoC

Diagrama de arquitectura del software:

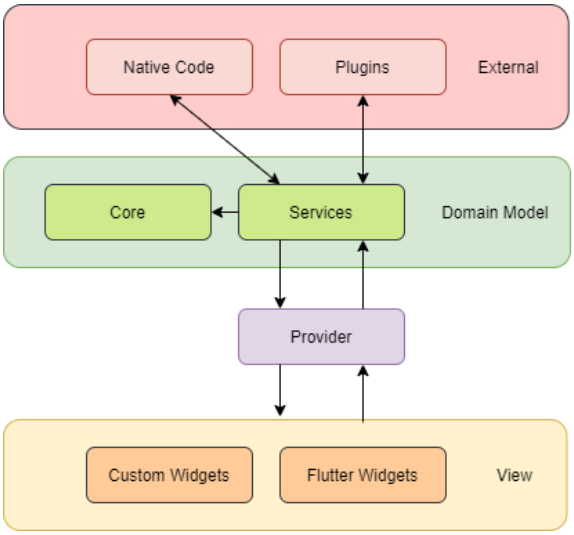

Figura 4: *Diagrama de arquitectura del software Spoca y Spoca.*

Como se puede ver en la figura 4, la arquitectura del software que presenta la aplicación se puede descomponer en las siguientes secciones:

View: El componente View contiene todo el código relacionado con la UI. Las diferentes pantallas y los diferentes widgets.
Model Domain: Este componente contiene la lógica de la aplicación.

- **Core:** Contiene los modelos de clases y el resto de las clases necesarias.
- **Services:** Este componente contiene las clases necesarias para realizar llamadas externas y recibir datos del componente External. Esta clase no se preocupa de la UI o de cómo la UI va a mostrar dichos datos.

Provider: Contiene el proveedor que conecta la UI con la lógica de la aplicación.
External: El componente External contiene paquetes de Dart, plugins externos, entre otros.

7.3.3.- Modelado de datos

En cuanto al modelado de datos, cabe destacar el uso de Firebase como Api Rest. Desde ésta, la aplicación realiza llamadas para obtener la diferente información almacenada en la base de datos que proporciona Firebase. Debe mencionarse que el uso de dichas llamadas se haya en la obtención de tipos de robots, las preferencias de limpieza del usuario, en los datos captados por los sensores los cuales son almacenados tras cada nueva detección y en la configuración de la piscina.

Por otra parte, el modelado de datos que se utiliza actualmente sobre los robots de piscina consiste en escrtucturas de diccionarios (clave-valor) que contienen

conjuntos de datos ficticios. Dichos datos proveen a la aplicación al arrancar de cierta información para poder simular lo que sería el normal uso de ésta. Ciertas funcionalidades permiten la gestión de los datos anteriormente mencionados, y de los nuevos que el usuario pueda crear a partir de otras funcionalidades. Cabe destacar que, una vez cerrada la aplicación, ésta perderá todas las acciones que se hayan realizado con los robots.

7.3.4.- Desarrollo de las funcionalidades

Multilenguaje:

Como punto de partida para el desarrollo, se ha partido de la idea de que crear la infraestructura para el multilenguaje facilitará el posterior desarrollo de la aplicación. Gracias a disponer desde el principio de la modificación de idioma, se puede internacionalizar de primeras todos los textos, lo que ahorra tiempo futuro en modificar estos.

Como se puede observar en la figura 5, SPOCA se ha creado para disponer de tres idiomas: inglés, español y catalán. Se ha tenido que hacer una adaptación de la propuesta que facilita Flutter para la internacionalización debido al hecho que esta app permite el cambio de idioma por interacción del usuario y no sólo dependiendo de la región, como ocurre en el caso de la propuesta oficial de Flutter.

Figura 5: *Captura modificación de idioma*

Navegación entre pantallas:

Para facilitar la navegación entre las pantallas principales se ha implemenetado un menú de *tabs* (figura 6) desde donde se puede acceder a estas.

Figura 6: *Captura menú de tabs*

Además de dicho menú inferior, la aplicación cuenta con otro menú lateral (figura 7) mediante el cual se puede acceder a pantallas secundarias como la modificación del idioma o la configuración de la piscina.

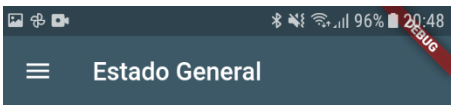

Figura 7: *Captura menú lateral*

Información de la piscina:

Dado que la PoC de SPOCA únicamente contempla la conexión con una supuesta base de carga de éstas, dicha base sólo enviará datos del estado actual de la piscina. Entre estos datos encontramos la temperatura del agua y el nivel de pH. Dentro de la pantalla de configuración de la piscina, como se puede ver en la figura 8, el usuario puede modificar valores límites que indiquen a la aplicación si los valores recibidos por la base son correctos o, por si lo contrario, la aplicación debe avisar al usuario para que tome las medidas necesarias.

La información proporcionada por la base de carga se muestra en la pantalla inicial de la app. Como se puede observar en la siguiente figura (figura 9), dicha información se muestra en diferentes tarjetas. Se puede apreciar lo comentado anteriormente sobre el *feddback* de los niveles de pH y la temperatura. Si los valores recibidos se hayan entre los límites establecidos por el usuario, la tarjeta correspondiente se muestra de color verde. Por lo contrario, si el valor no se encuentra entre dicho rango, la tarjeta se muestra de color rojo.

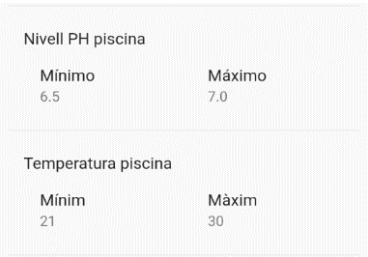

Figura 8: *Captura configuración de niveles*

Figura 9: *Captura cards pantalla de inicio*

Preferencias de limpieza:

En la pantalla de preferencias de limpieza, la cual se puede ver en la figura 10, el usuario puede modificar estas preferencias según su conveniencia. Cabe destacar que el código de la aplicación contempla dichas preferencias, pero estas sólo han sido incluidas como parte de la PoC, por lo que no tienen funcionalidad real. En el contexto de un proyecto comercial, estas preferencias serían enviadas al robot que sería capaz de interpretarlas y actuar en función de estas.

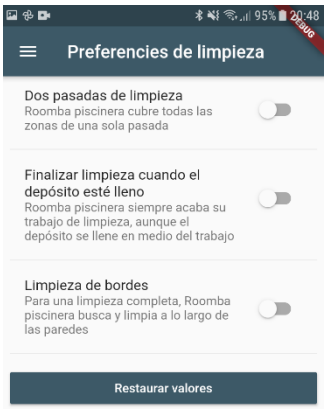

Figura 10: *Captura pantalla preferencias de limpieza*

App responsive:

Para el correcto desarrollo de esta PoC no se ha tenido unicamente en cuenta que la aplicación pueda ser *adaptive*, si no que tambien se ha tenido en cuenta que sea *responsive*. Para lograrlo, todas las pantallas dependen del tamaño del dipositivo que las ejecuta y no de tamaños fijos.

Cabe destacar que ciertas pantallas se han modificado según la orientación del dispositivo para su correcta visualización, como puede ser la pantalla de detalles del dispositivo.

En el caso concreto de la pantalla inicial, se ha considerado que el diseño que presenta con el dispositivo en vertical (figura 11) no se adapta adecudamente al tipo de orientación *landscape*. Para solucionar dicho problema se ha decidido utilizar un plugin externo *carousel_slider* [17] que nos permite darle otro aspecto más adecuado a la información de esta pantalla en esta orientación (figura 12).

Figura 11: *Captura pantalla inicial en portrait*

Figura 12: *Captura pantalla inicial en landscape*

Gestión de dispositivos:

Las funcionalidades que permiten la gestión de los dispositivos se encuentran distrubuidas entre diferentes pantallas de la aplicación.

Cabe destacar la vista de los dispositivos actuales introducidos por el usuario. En esta se muestra una lista con los diferentes dispositivos (figura 13), así como su estado: en uso, parado o sin conexión. Además, se permita la inserción de nuevos dispositivos mediante un formulario de pantallas dónde escoger el tipo de robot que se quiere insertar, realizar su correcta conexión entre la aplicación y el dispositivo, y la personalización de otro datos como el nombre o la programación semanal de este.

Figura 13: *Captura lista de dispositivos*

Otra vista a destacar es la de detalles del dispositivo (figura 14). En esta se puede ver toda su información, se permite modificar la programación del robot, así como conectarse con su correspondiente base de carga.

Figura 14: *Captura detalles del dispositivo*

Conectividad BLE:

Con el fin de dotar a la aplicación de SPOCA de conectividad BLE se ha procedido a adaptar parte de dicha lógica podemos encontrar las funcionalidades de descubrir dispositivos cercanos, conectar a un dispositivo específico, descubrir los servicios que este dispositivo ofrece y leer de éstos.

Como se ha comentado previamente, para la realización de esta PoC se ha empleado el dispositivo PAN1780 de Panasonic para simular la funcionalidad que tendría un robot SPOCA. Debido a que no está dentro del alcance de este proyecto desarrollar un firmware para dicha placa que simule fielmente el comportamiento de una SPOCA, se ha decidido

adaptar un firmware existente proporcionado por los desarrolladores de la placa.

Este firmware permite simular que dicha placa sea un dispositivo ble de *health rate*, con todos los servicios y la lógica que ello conlleva. Se ha adaptado dicha simulación para que envíe por uno de sus servicios periódicamente un valor que varía entre 10, 22 y 26 continuamente. Este valor representa los datos que enviaría el robot de sus sensores. Para facilitar el proyecto, dicho dato se utiliza para representar la temperatura actual de la piscina y el nivel de ph de ésta (dividiendo el valor entre 3 para conseguir un valor de ph que se aproxime a la realidad).

En la figura 15 se puede ver unas de las pantallas que conforman la creación de un nuevo dispositivo dentro de la aplicación. Dicha pantalla permite mostrar al usuario los dispositivos con conectividad BLE cercanos y conectarse a éste.

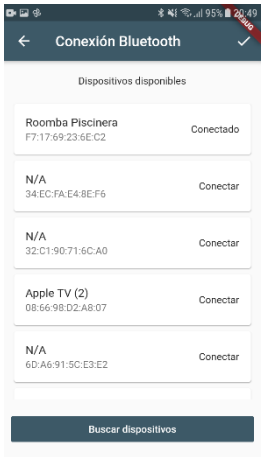

Figura 15: *Captura pantalla conexión bluetooth*

Debido a que el dispositivo donde se encuentre la aplicación puede llegar a salir del radio de acción de la conectividad BLE, dentro de la vista de *detalles del dispositivo* el usuario se puede acceder a la siguiente pantalla (figura 16) desde la cual se permite controlar el estado de la conexión con el robot SPOCA. Desde esta pantalla el usuario puede volver a conectarse con éste para volver a dotar a la aplicación de las funcionalidades que esta ofrece al estar conectado a un robot.

Figura 16: *Captura pantalla conectar dipositivo*

8. TESTEO

Siguiendo la filosofía de Idneo y pese a haber desarrollado simplemente una PoC, se ha querido crear una *Test Suit* dónde recoger una lista de tests funcionales que debe pasar la aplicación. Adjunto en el apéndice 3, se puede encontrar la tabla empleada con los resultados obtenidos en cada iteración de testeo. Por regla general, las aplicaciones desarrolladas por la empresa deben contener un conjunto de tests funcionales, probando así dichas aplicaciones en busca de *bugs* o *issues*.

9. RESULTADOS

A continuación se adjuntan dos enlaces que permiten visualizar sendos videos que muestran los resultados obtenidos de ambas aplicaciones, tanto de la APP_Template como de la aplicación de SPOCA.

- **APP_Template:**
  https://www.youtube.com/watch?v=pKWpm0RmkGA

- **SPOCA:**
  https://www.youtube.com/watch?v=QOLZzahQheA

10. CONCLUSIONES

10.1.- Conclusiones generales

Una vez finalizado el trabajo fin de grado se puede concluir que el resultado del proyecto es, en general, bastante satisfactorio. Basándonos en los resultados obtenidos, creo que el hecho de aprender una nueva tecnología y utilizarla para crear una aplicación ha dado mejores resultados de los esperados. Se ha conseguido que la aplicación posea todas las principales funcionalidades esperadas, por tanto, considero que es una buena PoC que poder enseñar a posibles clientes que puedan querer una aplicación multiplataforma.

Esta aplicación muestra fielmente los nuevos conocimientos que ha adquirido recientemente la empresa con la realización de este proyecto, tanto en el desarrollo de aplicaciones en Flutter como el conocimiento y el uso de BLE en esta tecnología.

10.2.- Conclusiones de aprendizaje

Bajo mi punto de vista, han sido bastante los nuevos conocimientos adquiridos durante la realización de este proyecto. Empezar con un nulo conocimiento en Flutter y poder terminar desarrollando la aplicación resultante creo que demuestra lo mucho que he podido aprender haciendo este TFG.

Creo que merece especial atención por otro lado, mencionar el hecho de que haber cursado ciertas asignaturas del grado de Ingeniería Informática ha servido de ayuda a la hora de realizar este proyecto. Es el caso

por ejemplo de de Ingenería del Software, de los cuales se ha extraído la formación para realizar los diagramas empleados en este proyecto y todo lo relacionado con las diferentes fases del diseño; Diseño del Software, para la realización de los mockups y la creación de la arquitectura del código; Requisitos del Software, que ha permitido tener el conocimiento correcto para poder reconocer y recopilar los requisitos propuesto por el cliente; entre otras asignaturas.

10.3.- Soluciones a problemas encontrado durante el proyecto

Pese a que, como he comentado anteriormente, no pudiera disponer del dispositivo final y de que el microprocesador empleado fuera recibido a mediados de diciembre, creo que el desarrollo de la aplicación es del todo favorable. En cierta forma, el hecho de no disponer de un dispositivo compatible con BLE hasta diciembre, ha obligado a modificar los tiempos previstos de este proyecto. Ciertas partes del desarrollo previstas para el principio del proyecto se tuvieron que ver atrasadas y a la inversa.

10.4.- Trabajo futuro

Me gustaría concluir este documento haciendo referencia a los posibles pasos que me gustaría poder realizar o que se deberían de realizar en el futuro.

Es cierto que me hubiera gustado disponer de más tiempo para poder llevar a cabo algunas de las siguientes partes descritas a continuación, pero considero que el trabajo realizado ha estado el acorde con el tiempo disponible para la realización de este TFG.

Disponer de un backend al cual poder realizar llamadas y obtener los datos de cada usuario sería uno de los primeros futuros pasos a realizar. Al haberse tenido en cuenta el posible uso de dichas llamadas desde el inicio del proyecto, el código se ha desarrollado para la fácil adaptación de estas.

Al ser una app multiplataforma, y dado que Flutter nos permite de una forma tan sencilla adaptar nuestra UI según el sistema operativo empleado, uno de los siguientes pasos a seguir sería la adaptación de la UI para los dispositivos iOS. Cabe decir que la aplicación actual puede utilizarse diretectamente en dispositivos con este sistema operativo, pero deberían utilizarse los elementos gráficos, widgets en el caso de Flutter, específicos de cada SO.

Creo que el principal paso a realizar posteriormente es el uso de un dispositivo SPOCA con el que poder desarrollar el código definitivo y adaptar la parte de código relacionada con la funcionalidad BLE.

Finalmente, sería interesante el hecho de habilitar una nueva funcionalidad que permitiera poner en marcha el robot y controlar el uso de este.

Agradecimientos

Finalmente, me gustaría concluir este documento dando las gracias principalmente a mi tutor Jorge Bernal por todo el apoyo ofrecido durante el transcurso de este. A su vez, agradecer a Idneo y a Daniel Bujalance por la oportunidad brindada, y a Maria Dolores Mimó por toda la ayuda prestada.

En último lugar, agradecer a mi familía y amigos por mostrarme su soporte durante la realización de este TFG.

Bibliografía

[1] Flutter,
https://flutter.dev/
[2] Idneo,
https://www.idneo.com/es/
[3] Prueba de concepto,
http://abancainnova.com/es/opinion/que-es-una-prueba-de-concepto/
[4] Dart,
https://dart.dev/
[5] Flutter. Showcase,
https://flutter.dev/showcase
[6] Ebay Motors App,
https://pages.ebay.com/motors/motorsapp/
[7] Visual Studio Code,
https://code.visualstudio.com/docs
[8] Git,
https://git-scm.com/
[9] Wikipedia. Bitbucket,
https://es.wikipedia.org/wiki/Bitbucket
[10] Android Studio,
https://developer.android.com/studio
[11] Firebase,
https://firebase.google.com/
[12] Kanabn,
https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban
[13] Udemy. Flutter & Dart - The Complete Guide [2020 Edition]
https://www.udemy.com/course/learn-flutter-dart-to-build-ios-android-apps/
[14] Panasonic. PAN1780
https://industry.panasonic.eu/devices/wireless-connectivity/bluetooth-low-energy-modules/pan1780-high-performance-and-long-range
[15] Pub. Flutter Reactive BLE
https://pub.dev/packages/flutter_reactive_ble
[16] Pub. Flutter Blue
https://pub.dev/packages/flutter_blue
[17] Pub. Carousel_slider
https://pub.dev/packages/carousel_slider

APÉNDICE

A1. DIAGRAMA DE GANTT

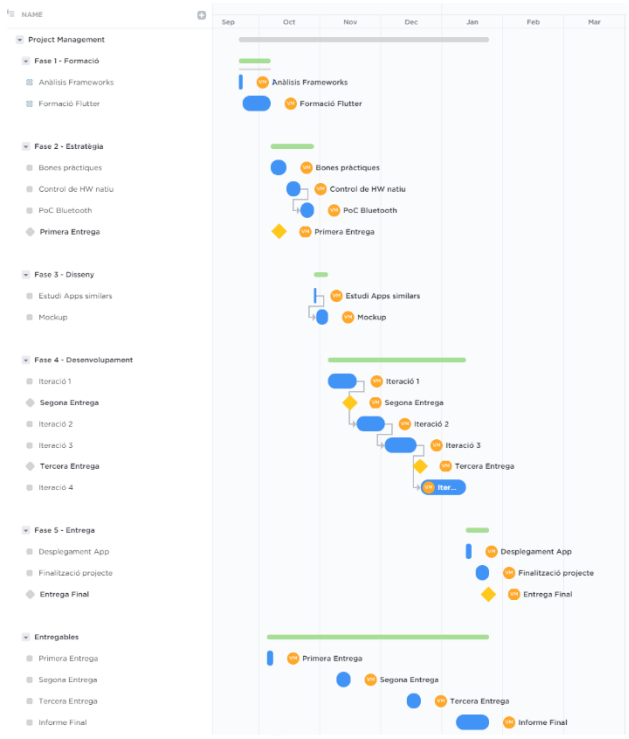

Figura 17: *Diagrama de Gantt*

A2. REQUISITOS TEMPLATE_APP

Requisitos funcionales

- La aplicación tiene que permitir escanear los dispositivos ceranos.
- La aplicación tiene que mostrar los dispositivos escaneados.
- La aplicación tiene que poder conectarse a un dispositivo específico.
- La aplicación tiene que poder desconectarse de un dispositivo específico.
- La aplicación tiene que poder descubrir los servicios de un dispositivo al que está conectado.
- La aplicación tiene que poder descubrir las características de un dispositivo al que está conectado.
- La aplicación tiene que poder descubrir los descriptores de un dispositivo al que está conectado.
- La aplicación debe mostrar todos los servicios descubiertos.
- La aplicación debe notificar cualquier cambio en el *status* del dispositivo.
- La aplicación debe poder leer características.
- La aplicación debe poder leer descriptores.
- La aplicación debe poder escribir descriptores.
- La aplicación debe poder escribir características.
- La aplicación tiene que ser multi lenguaje.
- La aplicación tiene que permitir realizar diagnósticos del dispositivo utilizando BLE.

Diagrama de casos de uso

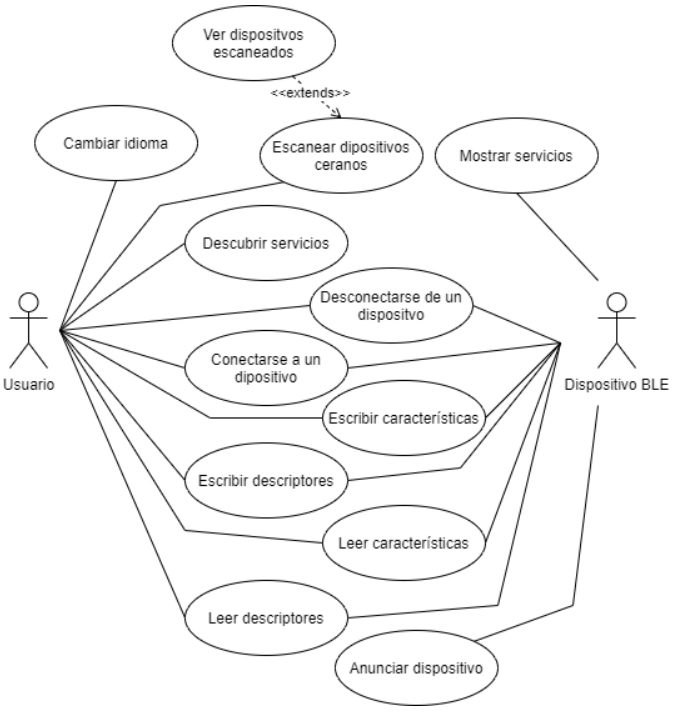

Figura 18: *Diagrama de casos de uso BLE_Template*

Requisitos no funcionales

- El sistema debe desarrollarse aplicando buenas prácticas de programación recomendadas por Dart.
- El sistema debe desarrollarse aplicando patrones de arquitectura recomendada por Flutter.
- La aplicación móvil debe conectarse al módulo de carga a una distancia mayor a 5 metros mediante el uso de Bluetooth.
- La aplicación debe utilizar elementos visuales estandarizados para su UI.
- El sistema tiene que estar desarrollado para plataformas Android.
- El sistema deberá estar desarrollado para plataformas IOS.
- El sistema tiene que contar con pruebas unitarias y de componentes.

A3. TEST SUIT

Test Case ID	Descripción Test Case	Resultado esperado	v0.0.1 (dev)	v0.1.1 (dev)	v0.1.2
TC1	Modificar el idioma de l'aplicación	Modificar el idioma, haciendo que todos los textos de la aplicación se modifiquen al idioma correspondiente	OK	OK	OK
TC2	Guardar el idioma seleccionado por el usuario	Al volver a abrir la aplicación, restaurar el idioma seleccionado por el usuario anteriorment	FAIL	OK	OK
TC3	Recuperar datos última detección	Al iniciar la aplicación, debe verse en la pantalla incial las diferentes cards con los últimos valores obtenidos por el sensor	FAIL	FAIL	OK
TC4	Card pH en rojo	Si el valor de pH recibido mediante ble no se encuentra entre los valores fijados por el usuario, la card de Ph deber verse en rojo	OK	OK	OK
TC5	Card pH en verde	Si el valor de pH recibido mediante ble se encuentra entre los valores fijados por el usuario, la card de Ph deber verse en verde	OK	OK	OK
TC6	Card Temperatura en rojo	Si el valor de temperatura recibido mediante ble no se encuentra entre los valores fijados por el usuario, la card de temperatura deber verse en rojo	OK	OK	OK
TC7	Card Temperatura en verde	Si el valor de temperatura recibido mediante ble se encuentra entre los valores fijados por el usuario, la card de temperatura deber verse en verde	OK	OK	OK
TC8	Estado de la piscina incorrecto	Si el valor actual de nivel de Ph o de temperatura no se encuentran dentro de sus valores, el estado de la piscina debe mostrarse como incorrecto	FAIL	FAIL	OK
TC9	Estado de la piscina correcto	Si el valor actual de nivel de Ph y de temperatura se encuentran dentro de sus valores, el estado de la piscina debe mostrarse como correcto	FAIL	FAIL	OK
TC10	Recuperar datos preferencias de limpieza	Al abrir la pantalla de preferencias de limpieza, deben recuperarse los valores previos indicados por el usuario	FAIL	OK	OK
TC11	Modificar datos preferencias de limpieza	El usuario modifica los valores actuales	FAIL	OK	OK
TC12	Restaurar datos preferencias de limpieza	El usuario restaura los valores de fábrica de preferencias de limpieza, y estos se ponen todos a false	FAIL	OK	OK
TC13	Navegación entre tabs	El usuario navega entre las diferentes pantallas de los tabs	OK	OK	OK
TC14	Nevegación por el drawer	El usuario navega entre las diferentes pantallas del drawer	OK	OK	OK
TC15	Crear dispositivo nuevo	El usuario crea un nuevo dispositivo mediante los siguientes pases del TC15	FAIL	OK	OK
TC15.1	Seleccionar tipo de dipositivo	El usuario selecciona el tipo de dispositivo a crear. Esto se debe mostrar en detalles del dispositivo	OK	OK	OK
TC15.2	Buscar dispositivos cercanos	El usuario visualiza los dispositivos ble cercanos. Esto se debe mostrar en detalles del dispositivo	OK	OK	OK
TC15.3	Conectar con dispositivo cercano	El usuario se conecta un dispsitivo ble cercano	FAIL	OK	OK

TC15.4	Modificar selección de días de la programación semanal	El usuario modifica los valores prestablecidos de la programación semanal. Estos se deben mostrar en detalles del dispositivo	FAIL	OK	OK
TC15.5	Modificar selección de hora de activación de un dia	El usuario modifica los valores prestablecidos de la hora de activación. Estos se deben mostrar en detalles del dispositivo	FAIL	OK	OK
TC15.6	Introducir nuevo nombre de dispositivo	El usuario introduce el nuevo nombre de dispositivo. Esto se debe mostrar en detalles del dispositivo	OK	OK	OK
TC22	Ver dispositivos disponibles	En la pantalla de dispositivos se muestran los dispositivos creados previamente por el usuario	OK	OK	OK
TC23	Ver detalles de un dispositivo	En la pantalla de detalles de un dispositivo se muestrna los detalles de un dispositivo creado	OK	OK	OK
TC24	Modificar programación semanal	Mediante el botón disponible en la pantalla de detalles de un dispositivo, el usuario modifica el valor de la progrmación semanal actual de un dispositivo	OK	FAIL	OK
TC25	Modificar nombre de un dispositivo	Mediante el botón disponible en la pantalla de detalles de un dispositivo, el usuario modifica el nombre del dispositivo	OK	OK	OK
TC26	Conectar dispositivo	El usuario se conecta un dispsitivo disponible	FAIL	OK	OK
TC27	Desconectar dispositivo	El usuario se desconecta un dispositivo al que estaba conectado	FAIL	FAIL	OK
TC28	Visualizar cambios en la pantalla principal con una nueva detección	La pantalla principal muestra automaticamente los nuevos valores de una deteción	FAIL	OK	OK
TC29	Modificar nivel mínimo de pH	El usuario debe poder modificar el valor mínimo de nivel de ph. Este valor debe ser menor o igual al valor máximo	OK	OK	OK
TC30	Modificar nivel máximo de pH	El usuario debe poder modificar el valor máximo de nivel de ph. Este valor debe ser mayor o igual al valor mínimo	OK	OK	OK
TC31	Modificar nivel mínimo de temperatura de piscina	El usuario debe poder modificar el valor mínimo de temperatura de piscina. Este valor debe ser menor o igual al valor máximo	OK	OK	OK
TC32	Modificar nivel máximo de temperatura de piscina	El usuario debe poder modificar el valor máximo de temperatura de piscina. Este valor debe ser mayor o igual al valor mínimo	OK	OK	OK
TC33	La navegación entre pantallas debe ser correcta	El usuario debe poder navegar entre las diferentes pantallas de la aplicación acorde al mapa de pantallas	FAIL	FAIL	OK